What's New in Cocoa

Session 101

Ali Ozer
Manager, Cocoa Frameworks

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

What's New in Cocoa

- We will:
  - Cover changes in AppKit and Foundation since SnowLeopard
  - Provide pointers to other talks
- This is not "What's New in Cocoa Touch"

What's New in Cocoa Touch	Pacific Heights Thursday 4:30PM

Topics

- Auto Save and Versions
- Resume
- Automatic termination
- iCloud support
- Autolayout
- Aqua
- Text
- Other Topics

Auto Save and Versions

Auto Save and Versions

What's the Problem?

- Users have to explicitly save documents
- Users need to continue to save regularly
- Users are sometimes forced to save at inopportune times

Auto Save and Versions

- Eliminate the need to save explicitly
  - Save document automatically, and “always”
  - Save in place
- Make it easier to avoid unintentional changes
  - M nt is bein
  - M erhaps not
- Ma s

Item	Amount	
Travel Expenses	$4200	
Salaries	$300,000	
Taxes	$15	
Food	$6700	
Candy	$35,000	70,000
"Entertainment"	$5000	
Sales	$42	1
Other Income	$900,000	
TOTAL	**$559,127**	

2010 Budget

Item	Amount	
Travel Expenses	$5,200	
Salaries	$200,000	
Taxes	$156	
Food	$14,000	
Candy	$36,000	72,000
"Entertainment"	$10,000	
Sales	$84	2
Other Income	$300,000	
TOTAL	**$54,728**	

2011 Budget

Demo

NSDocument

- Provides Auto Save and Versions functionality out of the box
- Override:

```
+ (BOOL)autosavesInPlace {
    return YES;
}
```

NSDocument

Further To-Dos

- Asynchronous saving
- Versions UI customization
- Identifying “discardable” changes

File Coordination

A mechanism for locking and notification

- Enables serialization of document accesses
- Makes sure documents on disk are "up to date"
- New APIs
  - NSFileCoordinator class
  - NSFilePresenter protocol

Taking Advantage of File Coordination

Pacific Heights
Tuesday 4:30PM

NSFileCoordinator

Sample Reader

```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileCoordinator

Sample Reader

```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                   options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileCoordinator

Sample Reader

```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                   options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileCoordinator

Sample Reader

```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileCoordinator

Sample Reader

```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileCoordinator

Sample Reader

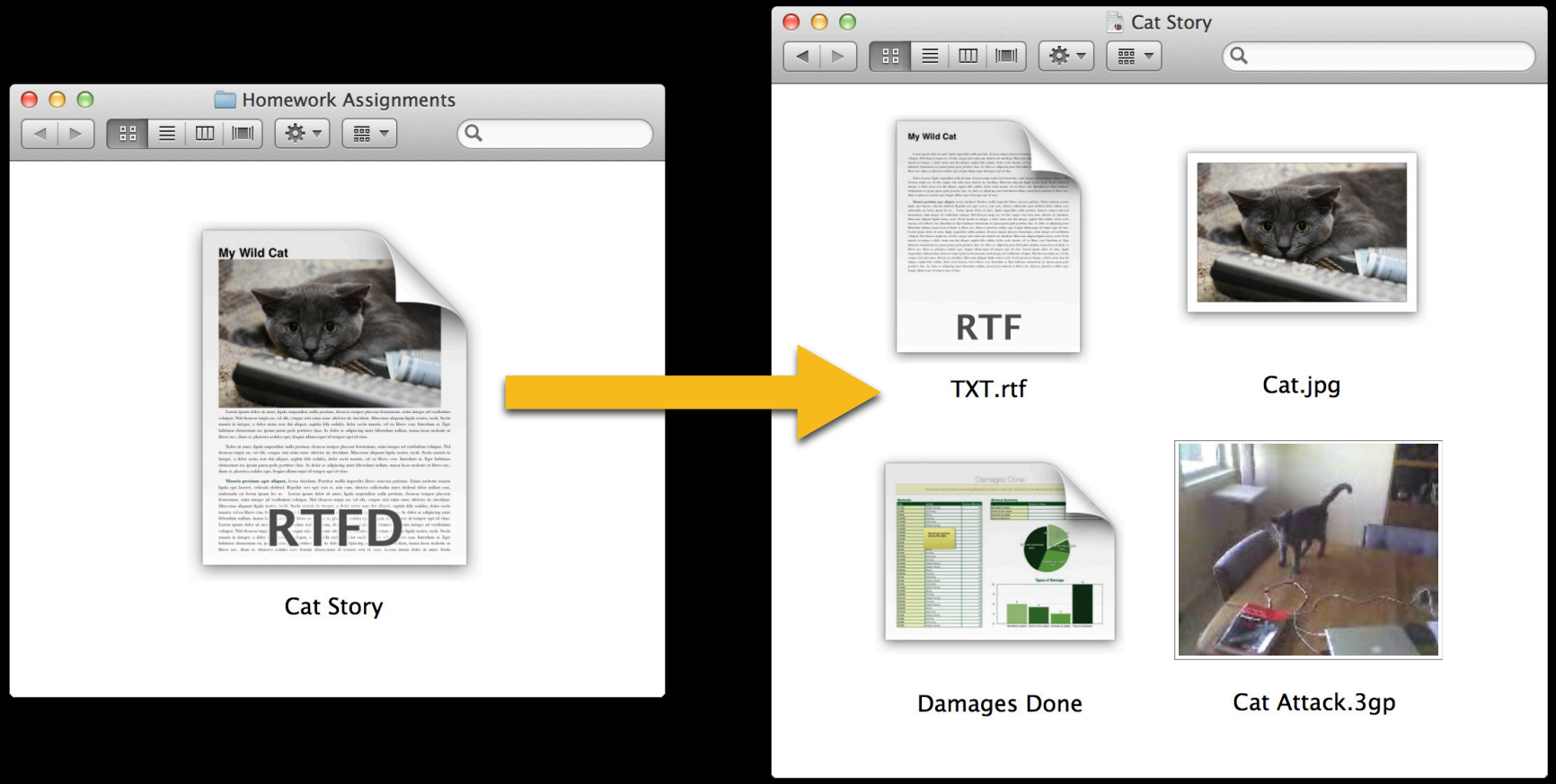
```
NSFileCoordinator *coordinator = [[NSFileCoordinator alloc] init];
__block NSError *error = nil;
__block NSData *data = nil;

[coordinator coordinateReadingItemAtURL:url options:0 error:&error
                byAccessor:^(NSURL *newURL) {
                     data = [[NSData alloc] initWithContentsOfURL:newURL
                                                options:0 error:&error];
                }];

if (data) ...    // Success; deal with data and release it when done
else ...         // Error
```

NSFileWrapper

A fine class for efficiently managing file packages



NSFileWrapper

A fine class for efficiently managing file packages

- Now in Foundation
- Also available in iOS (as of 4.0)

Resume

a.k.a. State Restoration

Resume

- Make restart, shutdown, and logout seamless
  - Eliminate need to save changes
  - Eliminate need to recreate state
- Also use same technology to make Quit preserve state

Resume

- On login, relaunch apps that were running
  - Apps can control whether they are relaunched
- Provide mechanism for apps to save and restore state
  - Automatic for a number of cases
  - APIs provided for customization

Resume and Automatic Termination in Lion	Marina Thursday 9:00AM

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [super encodeRestorableStateWithCoder:coder];
    [self encodeInteger:[self selectedItem] forKey:@"SelectedItem"];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [super encodeRestorableStateWithCoder:coder];
    [self encodeInteger:[self selectedItem] forKey:@"SelectedItem"];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [super encodeRestorableStateWithCoder:coder];
    [self encodeInteger:[self selectedItem] forKey:@"SelectedItem"];
}
- (void)restoreStateWithCoder:(NSCoder *)coder {
    [super restoreStateWithCoder:coder];
    NSInteger item = [coder decodeIntegerForKey:@"SelectedItem"];
    [self setSelectedItem:item];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [super encodeRestorableStateWithCoder:coder];
    [self encodeInteger:[self selectedItem] forKey:@"SelectedItem"];
}
- (void)restoreStateWithCoder:(NSCoder *)coder {
    [super restoreStateWithCoder:coder];
    NSInteger item = [coder decodeIntegerForKey:@"SelectedItem"];
    [self setSelectedItem:item];
}
```

Resume

Example of saving and restoring custom state

```
- (void)setSelectedItem:(NSInteger)item {
    ...
    [self invalidateRestorableState];
}
- (void)encodeRestorableStateWithCoder:(NSCoder *)coder {
    [super encodeRestorableStateWithCoder:coder];
    [self encodeInteger:[self selectedItem] forKey:@"SelectedItem"];
}
- (void)restoreStateWithCoder:(NSCoder *)coder {
    [super restoreStateWithCoder:coder];
    NSInteger item = [coder decodeIntegerForKey:@"SelectedItem"];
    [self setSelectedItem:item];
}
```

Resume

Even simpler example, for KVC and KVO-compliant properties

```
+ (NSArray *)restorableStateKeyPaths {
    return [[super restorableStateKeyPaths]
                                    arrayByAddingObject:@"SelectedItem"];
}
```

Resume

Even simpler example, for KVC and KVO-compliant properties

```
+ (NSArray *)restorableStateKeyPaths {
    return [[super restorableStateKeyPaths]
                                    arrayByAddingObject:@"SelectedItem"];
}
```

Automatic Termination

Automatic Termination

What's the Problem?

- Users need to explicitly quit applications
- System cannot kill applications to reclaim resources

Automatic Termination

- Enable applications to declare that their process can be killed
- Applications should do this only if it can be done without the user noticing
  - App has no visible windows
  - App is not doing any background activity
  - App supports Resume

Demo

iCloud Support

iCloud Support

- APIs for synchronizing data through the cloud
  - Documents
  - Configuration data

iCloud Storage Overview	Presidio Tuesday 11:30AM
Auto Save and Versions in Mac OS X 10.7 Lion	Pacific Heights Tuesday 3:15PM

NSFileManager

Putting a document in the cloud

```
- (BOOL)setUbiquitous:(BOOL)flag
            itemAtURL:(NSURL *)url
       destinationURL:(NSURL *)destinationURL
                error:(NSError **)error;
```

NSFileManager

Interacting with local copies of documents in the cloud

```
- (BOOL)startDownloadingUbiquitousItemAtURL:(NSURL *)url
                                     error:(NSError **)error;

- (BOOL)evictUbiquitousItemAtURL:(NSURL *)url
                           error:(NSError **)error;
```

NSMetadataQuery

Enumerating documents in the cloud

```
NSString *NSMetadataQueryUbiquitousDocumentsScope;
NSString *NSMetadataQueryUbiquitousDataScope;
...
```

NSURL

Querying attributes of documents in the cloud

```
NSString *NSURLIsUbiquitousItemKey;

NSString *NSURLUbiquitousItemIsDownloadedKey;
NSString *NSURLUbiquitousItemIsDownloadingKey;
NSString *NSURLUbiquitousItemPercentDownloadedKey;

NSString *NSURLUbiquitousItemHasUnresolvedConflictsKey;

...
```

NSUbiquitousKeyValueStore

Storing configuration data in the cloud

```
@interface NSUbiquitousKeyValueStore : NSObject

- (id)objectForKey:(NSString *)aKey;
- (void)setObject:(id)anObject forKey:(NSString *)aKey;
- (void)removeObjectForKey:(NSString *)aKey;

- (BOOL)synchronize;

...
@end
```

Autolayout

Autolayout

- New view layout system in Cocoa
- Constraint-based

  - Examples
    - “Button1 is 12 points to the left of Button2”
    - “Button1 and Button2 have the same baseline”
    - “Button1 is at least 80 points wide”
    - “Button1 is standard Aqua spacing away from the edge of its container”

NSLayoutConstraint

“Button1 is 12 points to the left of Button2,” in code

```
id myConstraint = [NSLayoutConstraint
                     constraintWithItem:button1
                              attribute:NSLayoutAttributeRight
                              relatedBy:NSLayoutRelationEqual
                                 toItem:button2
                              attribute:NSLayoutAttributeLeft
                             multiplier:1.0
                               constant:-12.0];
```

NSLayoutConstraint

“Button1 is 12 points to the left of Button2,” in code

```
id myConstraint = [NSLayoutConstraint
                    constraintWithItem:button1
                             attribute:NSLayoutAttributeRight
                             relatedBy:NSLayoutRelationEqual
                                toItem:button2
                             attribute:NSLayoutAttributeLeft
                            multiplier:1.0
                              constant:-12.0];

[containerView addConstraint:myConstraint];
```

NSLayoutConstraint

Visual Format Language

```
NSDictionary *views = NSDictionaryOfVariableBindings(button1, button2);
id myConstraint = [NSLayoutConstraint
                    constraintsWithVisualFormat:@"[button1]-12-[button2]"
                                        options:0
                                        metrics:nil
                                          views:views];
```

NSLayoutConstraint

Visual Format Language

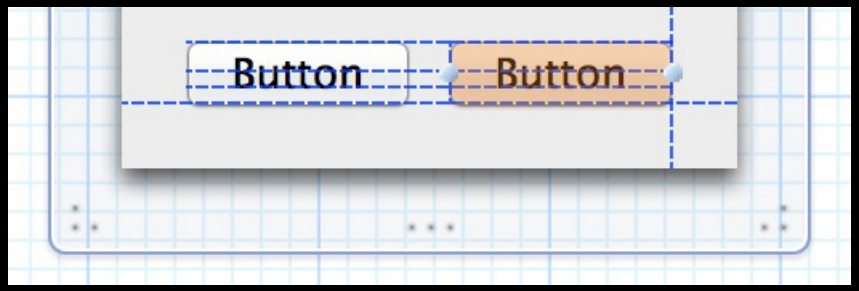
```
NSDictionary *views = NSDictionaryOfVariableBindings(button1, button2);
id myConstraint = [NSLayoutConstraint
                    constraintsWithVisualFormat:@"[button1]-12-[button2]"
                                        options:0
                                        metrics:nil
                                          views:views];
```

"[button1]-[button2]"

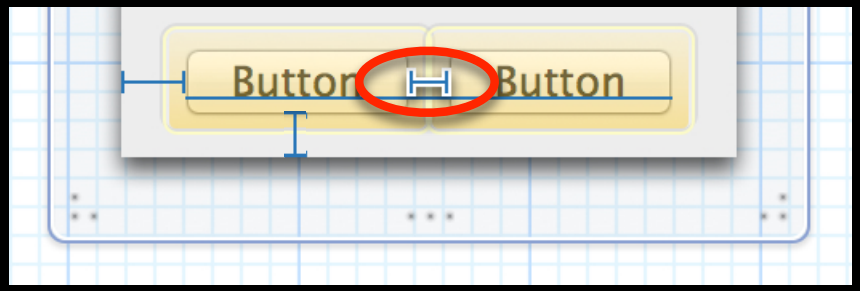
NSLayoutConstraint

Interface Builder

- Guides become persistent constraints

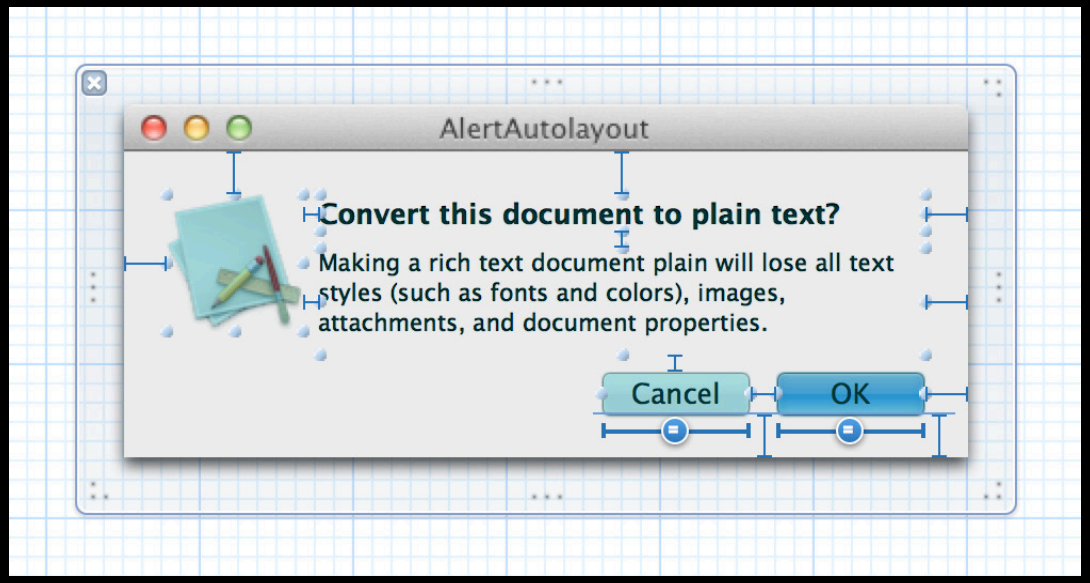


Guides

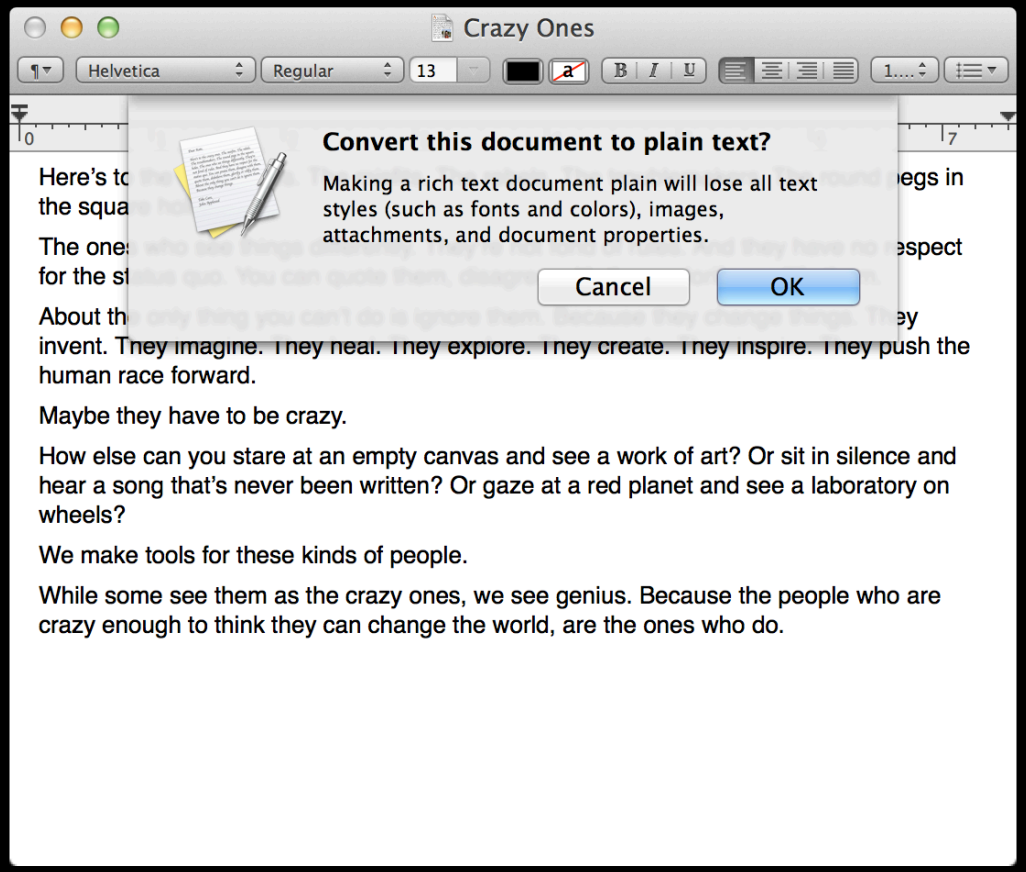


Constraints

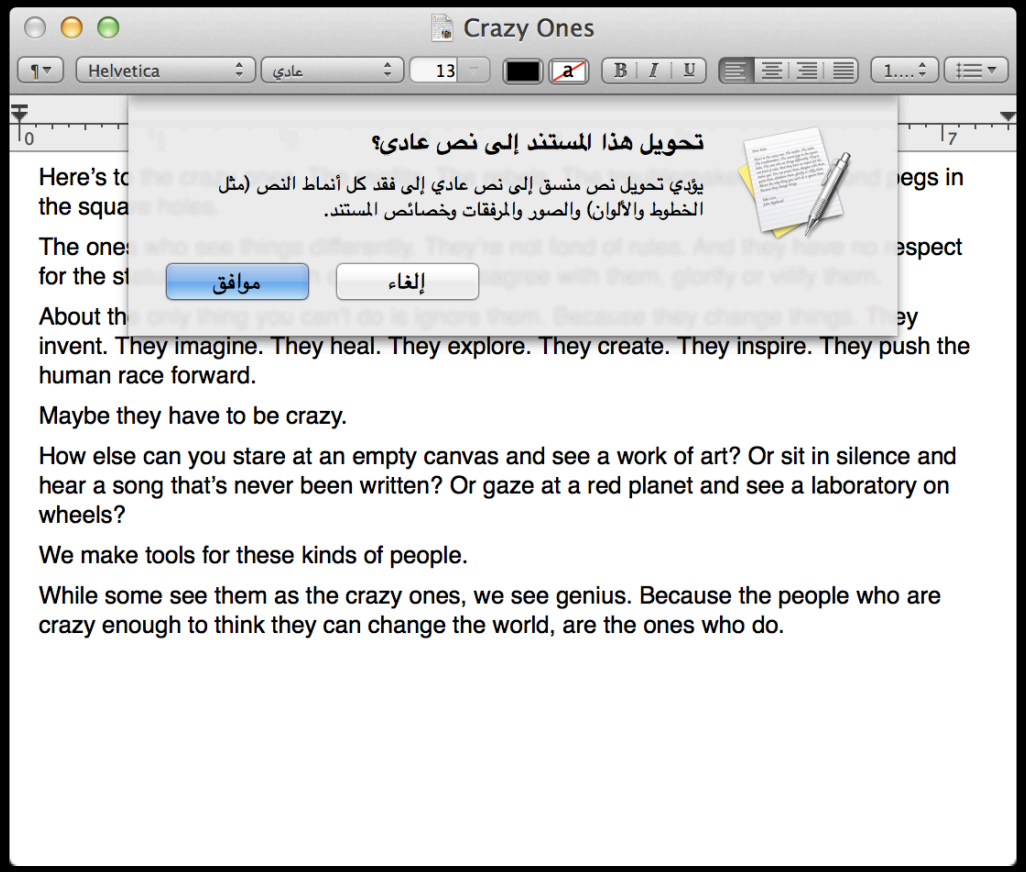
NSAlert with Autolayout



NSAlert in English



NSAlert in Arabic



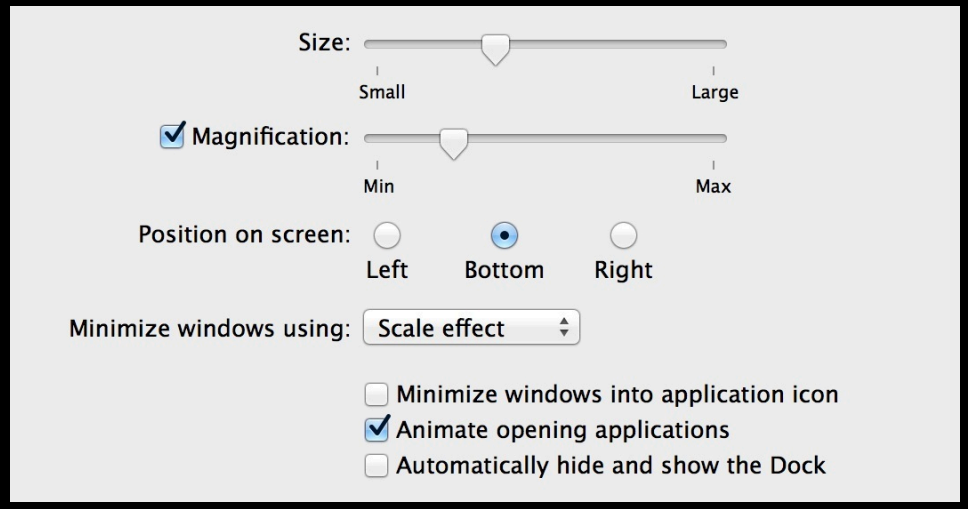
Aqua

Aqua

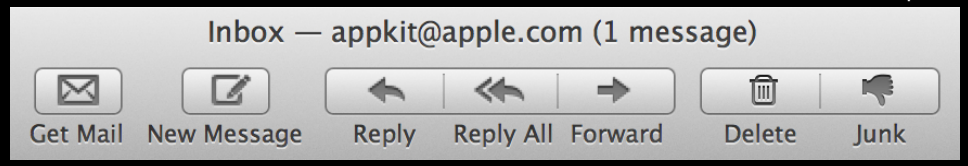
- Aqua
- Full screen
- Scrolling
- Multi-Image dragging
- NSPopover
- View-based NSTableView
- Resolution independence

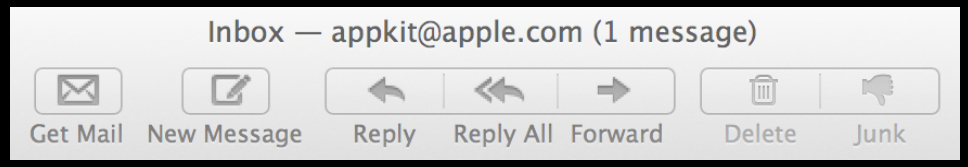
Aqua Changes

- New look
- Engraved text on controls
- Treatment of enabled controls in inactive windows



SnowLeopard





Full Screen

A new mode for working

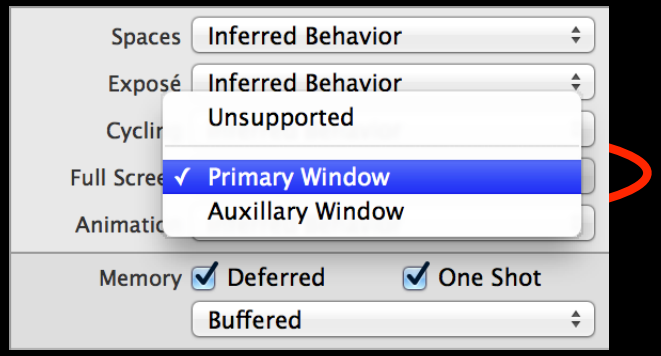
- Use as much of the screen as possible
- Reduce distractions
  - Hide, but still provide access to the Dock, menubar, toolbar, etc.
- Enable easy access to other apps
  - Mission Control
  - Swipe

Enabling Full Screen

- Specify which of your app's windows can go into full screen
  - In code

```
[win setCollectionBehavior:NSWindowCollectionBehaviorFullScreenPrimary];
```

  - In NSWindow inspector of Interface Builder
- Add menu item "Enter Full Screen"
- Optionally, customize:
  - Size
  - Animation
  - Layout

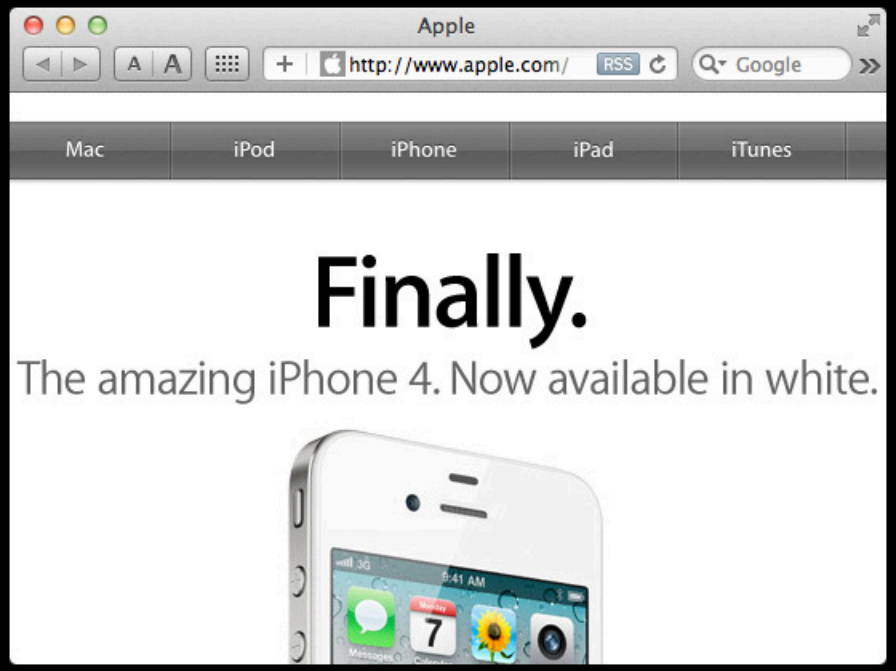


Scrolling Changes

- Inverted scrolling
- Elastic scrolling
- Fluid swipes
- Overlay scrollers

Overlay Scrollers

- Appear when needed
- Don't take space away from content



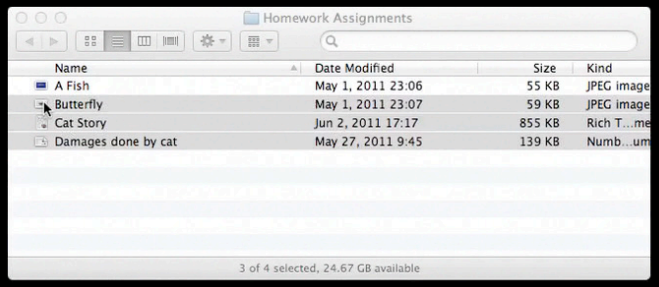
SnowLeopard

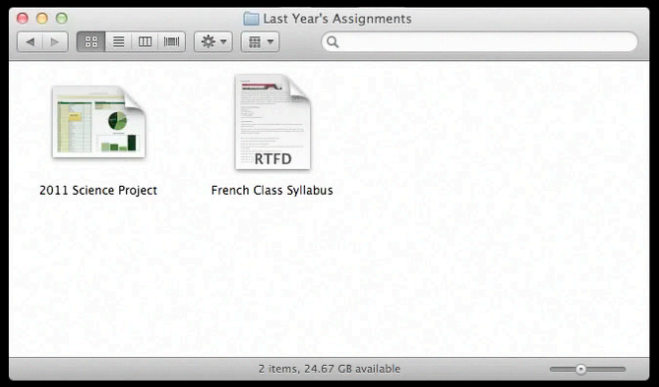
Multi-Image Dragging

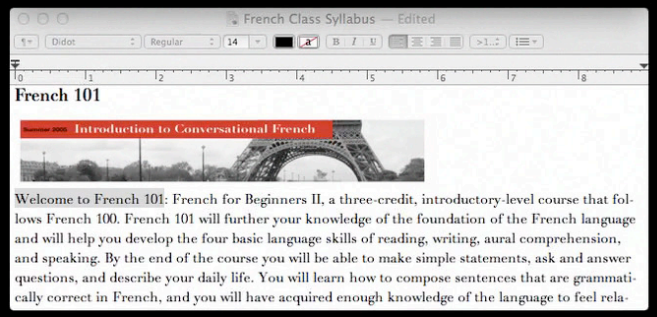
- Fine control over dragging multiple items
- Customizable look and flocking behaviors

Multi-Image Dragging

- Fine control over dragging multiple items
- Customizable look and flocking behaviors
- New classes
  - NSDraggingSession, NSDraggingItem, NSDraggingImageComponent
- Formalized protocols
  - NSDraggingSource, NSDraggingDestination





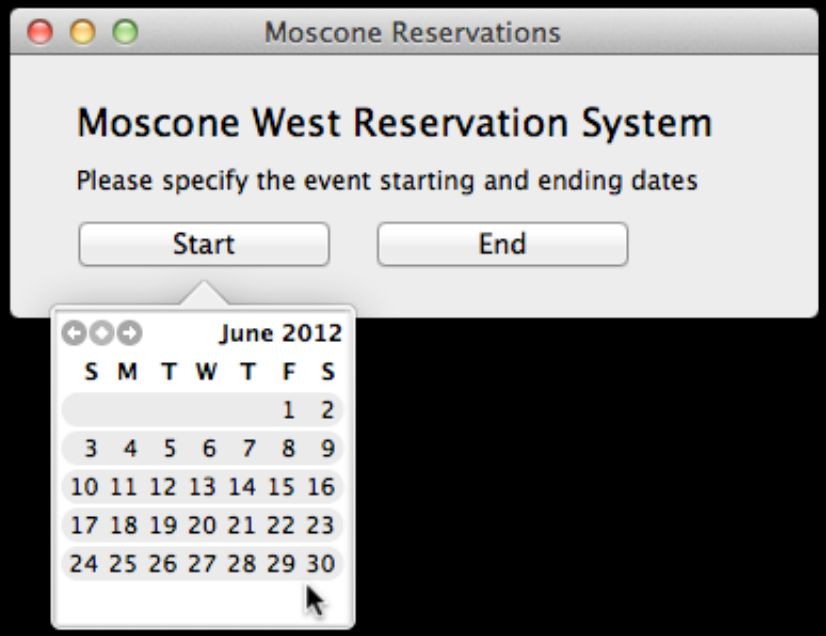


Scrolling, Swiping, Dragging: Now with More Animation!

Russian Hill
Wednesday 3:15PM

NSPopover

- A "popup" window for transient content
- Can host any view
- Can be detached



View-Based NSTableView

- NSTableView and NSOutlineView can now host NSViews
- New classes
  - NSTableRowView
  - NSTableCellView

Other NSTableView Changes

- Animations
- Multi-image dragging
- Floating group rows

View Based NSTableView Basic to Advanced	Nob Hill Thursday 10:15AM

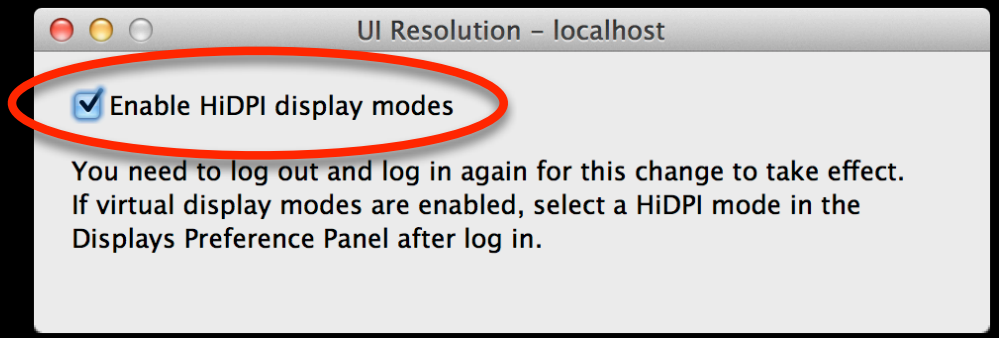
Resolution Independence

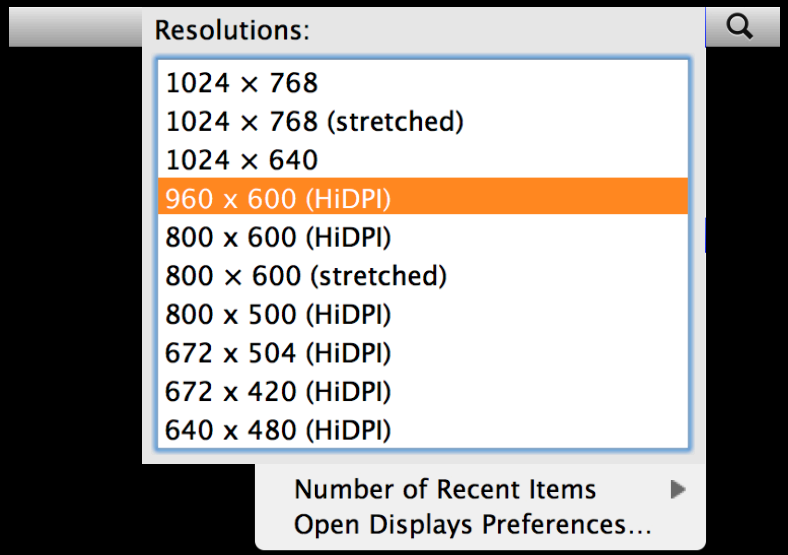
Architecture update

- Device-independent Quartz model extended to screens
  - Screen size and window frames are now in points
- Display can be configured for 2x resolution
  - Enable through Quartz Debug

Resolution Independence

Enabling 2x mode

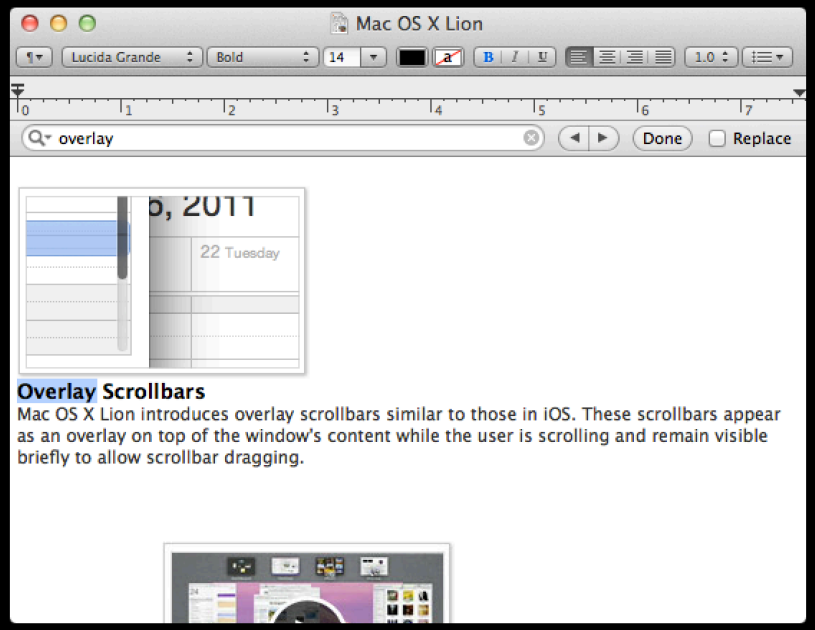




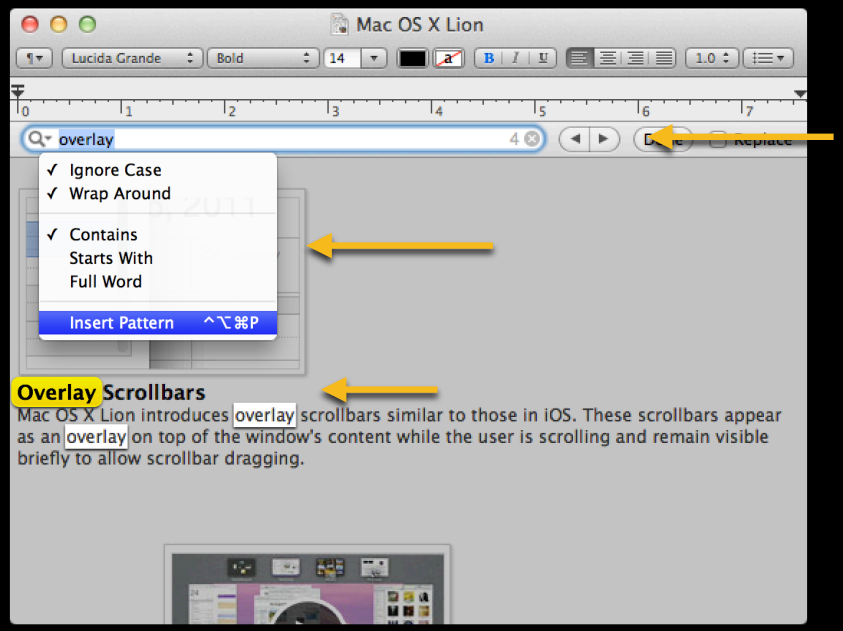
Text

Aki Inoue
Cocoa Senior Software Engineer

Text Finder


Mac OS X Lion
Lucida Grande
Bold
14
1.0
overlay
Done
Replace
22 Tuesday
Overlay Scrollbars
Mac OS X Lion introduces overlay scrollbars similar to those in iOS. These scrollbars appear as an overlay on top of the window's content while the user is scrolling and remain visible briefly to allow scrollbar dragging.

Text Finder



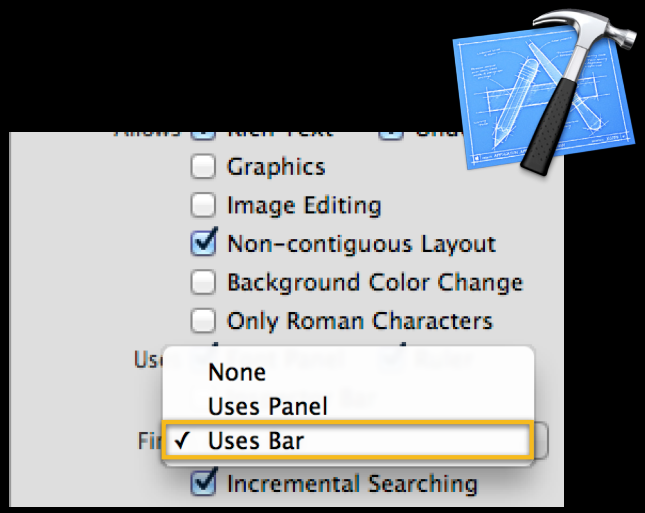
- Standard text searching UI
- Find bar support
- Incremental search
- Advanced search options
- Out-of-box with the Text System
- Works with non-NSTextView clients

Built-in Text System Support

Text Finder with NSTextView

- Enabling find bar

```
[yourTextView setUsesFindBar:YES];
```

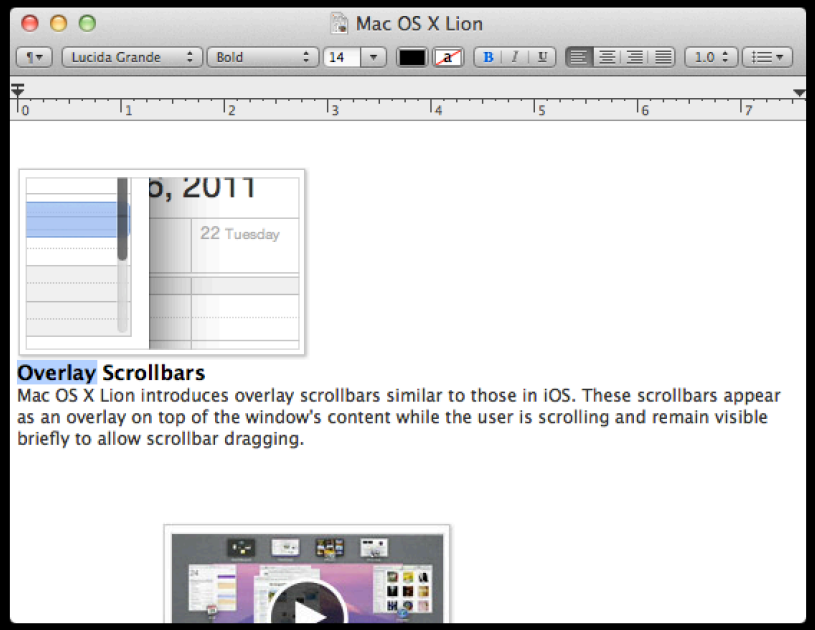


Working with Text Finder

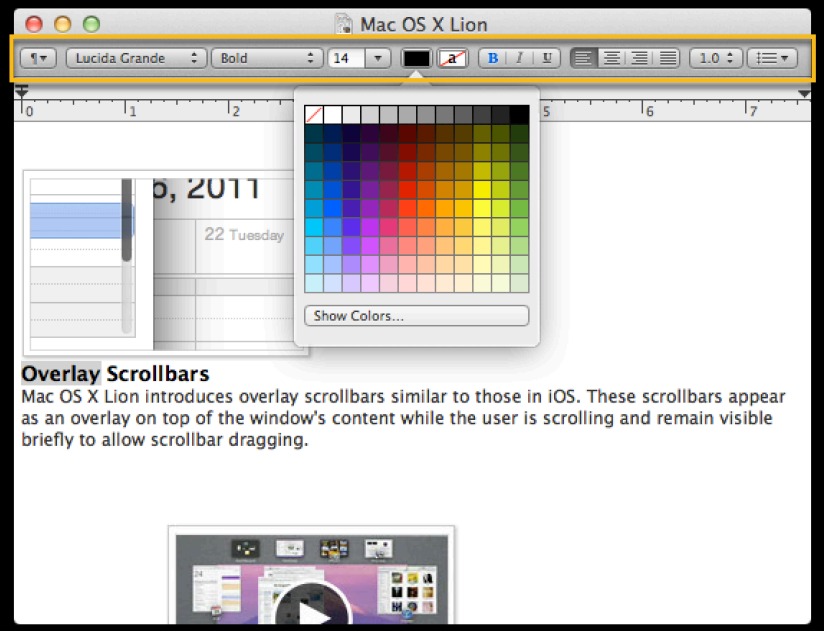
- 1 controller class & 2 protocols
  - NSTextFinder
  - NSTextFinderClient
  - NSTextFinderBarContainer

- Non-text view clients
- Custom find bar hosting view

Inspector Bar



Inspector Bar

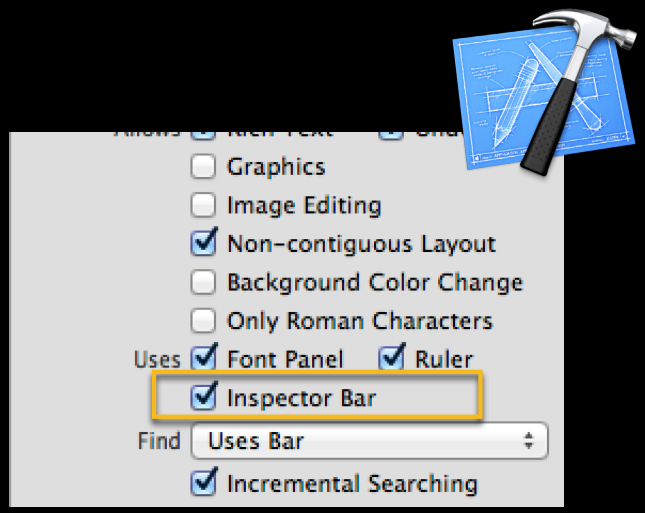


- Provides intuitive text controls
- Integrated into window title bar
- Many UI improvements
- Replaces the text accessory view

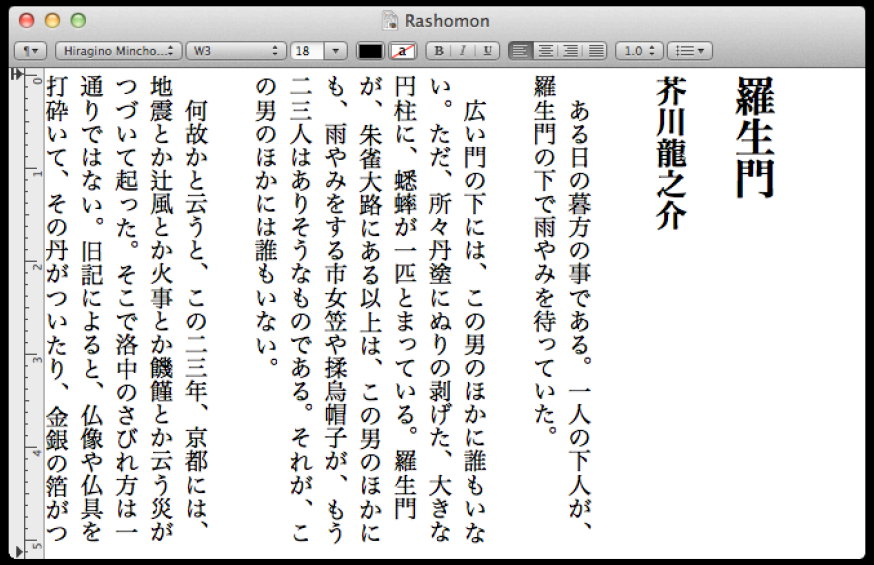
Inspector Bar

- Enabling inspector bar

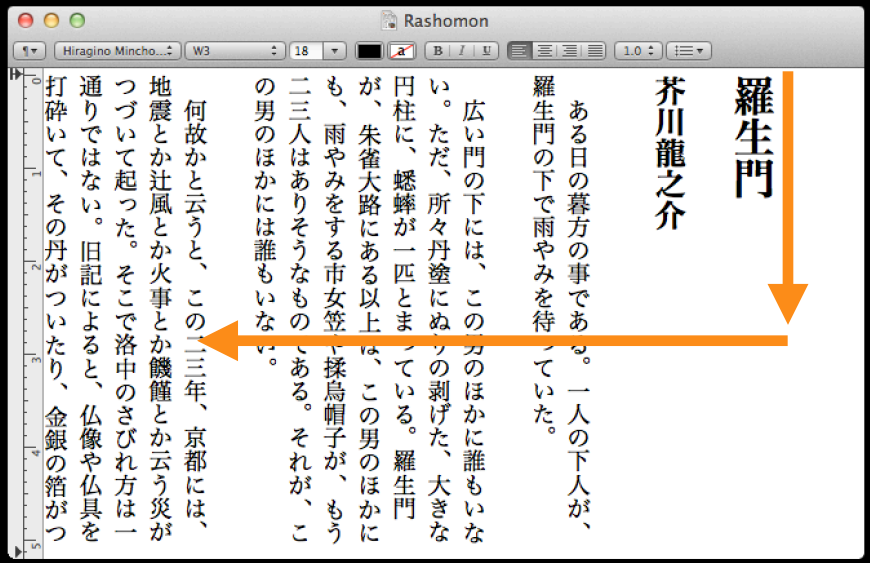
```
[yourTextView setUsesInspectorBar:YES];
```



Vertical Text



Vertical Text



- East Asian vertical text layout
- Top-to-bottom, right-to-left
- MVC role separation
- Open-ended architecture
- Per-container orientation

Configuring NSTextView Vertically

- Changing the layout orientation

```
[yourTextView setLayoutOrientation:NSTextLayoutOrientationVertical];
```

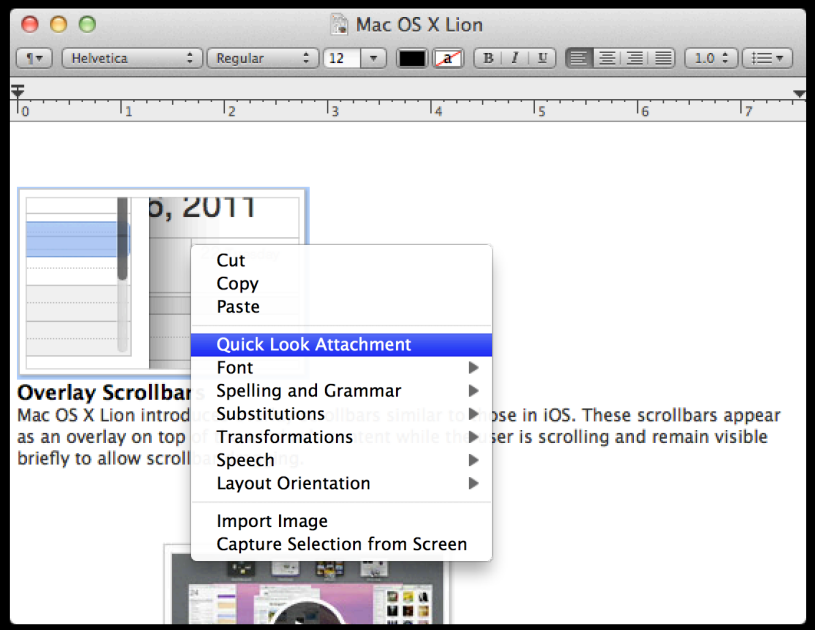
- Action method used by the context menu
  - -changeLayoutOrientation

- Additional NSTextView properties affected
  - Bounds rotation
  - Resize direction
  - Scroller visibility
  - Ruler configuration

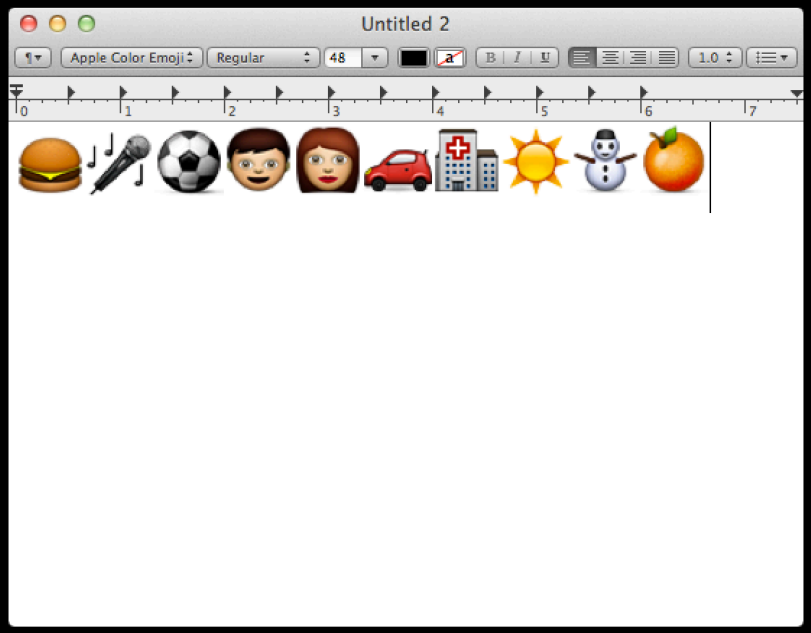
Demo

Text

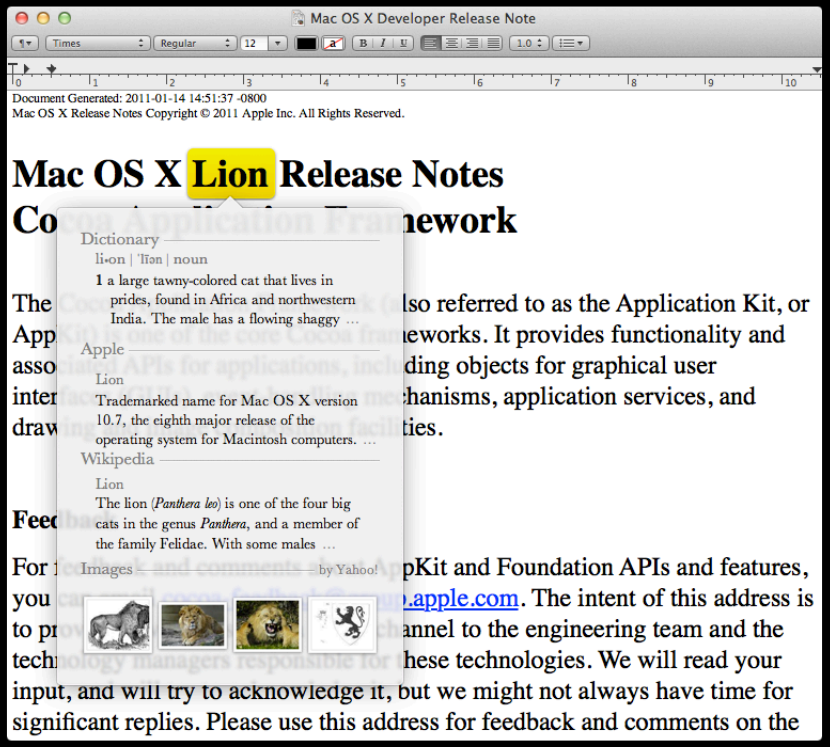
Quick Look


Mac OS X Lion
Helvetica
Regular
12
1.0
Cut
Copy
Paste
Quick Look Attachment
Font
Spelling and Grammar
Substitutions
Transformations
Speech
Layout Orientation
Import Image
Capture Selection from Screen
Overlay Scrollbar
Mac OS X Lion introd
as an overlay on top
briefly to allow scroll
ose in iOS. These scrollbars appear
ser is scrolling and remain visible

Color Emoji



LookUp


Mac OS X Developer Release Note
Document Generated: 2011-01-14 14:51:37 -0800
Mac OS X Release Notes Copyright © 2011 Apple Inc. All Rights Reserved.
Mac OS X Lion Release Notes
Dictionary
li•on | ˈlīən | noun
1 a large tawny-colored cat that lives in prides, found in Africa and northwestern India. The male has a flowing shaggy …
Apple
Lion
Trademarked name for Mac OS X version 10.7, the eighth major release of the operating system for Macintosh computers. …
Wikipedia
Lion
The lion (Panthera leo) is one of the four big cats in the genus Panthera, and a member of the family Felidae. With some males …
Images
by Yahoo!

Text Enhancements

- Text checking integration
- Font Collection
- Press and Hold accented character input
- and more…

Other Topics

Automatic Reference Counting

Let the compiler do retain/release/autorelease

- Works great in Cocoa on 10.7
  - Available for 10.6 apps but without zeroing weak references
- ARC code can load and work with manual reference counted code
- No change in the treatment outlets, delegates, targets
  - Continue to treat them as weak

Core Data

- Auto Save support in NSPersistentDocument
- iCloud support
- Ordered to-many relationships
- New concurrency APIs
- NSIncrementalStore
- Xcode 4 modeling tools

What's New in Core Data on Mac OS X	Nob Hill Thursday 11:30AM

Some Other Foundation Enhancements

- NSJSONSerialization
- NSOrderedSet
- NSURL keys for additional file and volume properties
- Keyed archiving support for Distributed Objects
- More precise removal of Key-Value observers
- String handling improvements
  - NSRegularExpression, NSDataDetector, NSLinguisticTagger
- Week based calendar improvements in NSCalendar

Performing Calendar Calculations	Russian Hill Wednesday 4:30PM

Some Other AppKit Enhancements

- Apple Push Notifications
- Much more Finder-like open/save panels
- Animated window opening/closing
- Support for application sandboxing
- Better debugging of exceptions
  - User default NSApplicationShowExceptions
- Accessibility

Summary

Many user and developer features in 10.7

- Auto Save
- Resume
- Automatic termination
- iCloud
- Full Screen
- Gestures
- Autolayout
- And many more!

Resources

- Release notes
  - AppKit ("Cocoa Application Framework")
  - Foundation ("Cocoa Foundation Framework")
- Header files
  - Search for "10_7"
    - MAC_OS_X_VERSION_10_7
    - NS_AVAILABLE_MAC(10_7), NS_AVAILABLE(10_7, ...)
    - NS_DEPRECATED_MAC(..., 10_7), NS_DEPRECATED(..., 10_7, ..., ...)
- Documentation

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Documentation
Mac OS X Dev Center
http://developer.apple.com/devcenter/mac

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Cocoa Autolayout	Marina Tuesday 11:30AM
Auto Save and Versions in Mac OS X 10.7 Lion	Pacific Heights Tuesday 3:15PM
Taking Advantage of File Coordination	Pacific Heights Tuesday 4:30PM

Related Sessions

Full Screen and Aqua Changes	Russian Hill Wednesday 10:15AM
Scrolling, Swiping, Dragging: Now with More Animation!	Russian Hill Wednesday 3:15PM
Performing Calendar Calculations	Russian Hill Wednesday 4:30PM

Related Sessions

Resume and Automatic Termination in Lion	Marina Thursday 9:00AM
View Based NSTableView Basic to Advanced	Nob Hill Thursday 10:15AM
What's New in Core Data on Mac OS X	Nob Hill Thursday 11:30AM
Advanced Text Processing	Mission Thursday 3:15PM
Design Patterns to Simplify Mac Accessibility	Pacific Heights Thursday 3:15PM

Labs

Cocoa Lab	App Frameworks Lab A Tuesday 2:00-6:00PM
Cocoa and AutoLayout Lab	App Frameworks Lab A Wednesday 9:00-11:00AM
Cocoa, Full Screen, and Aqua Lab	App Frameworks Lab A Wednesday 2:00-6:00PM
Cocoa, Auto Save, File Coordination, and Resume Lab	App Frameworks Lab A Thursday 2:00-4:00PM

WWDC2011